

FE-DeepONet:

A hybrid solver based on physics-informed deep operator networks for multiscale simulations

Hamidreza Eivazi¹, M. Alikhani¹, J.-A. Tröger², S. Wittek¹, S. Hartmann² and A. Rausch¹

¹Institute for Software and Systems Engineering, Clausthal University of Technology, Germany

²Institute of Applied Mechanics, Clausthal University of Technology, Germany

@ he76@tu-clausthal.de



Contents

Problem statement

Physics-informed neural networks (PINNs)

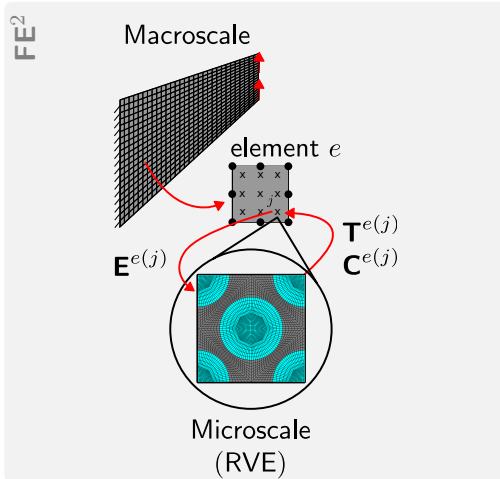
Deep operator networks (DeepONets)

Methodology: FE-DeepONet for multiscale simulations

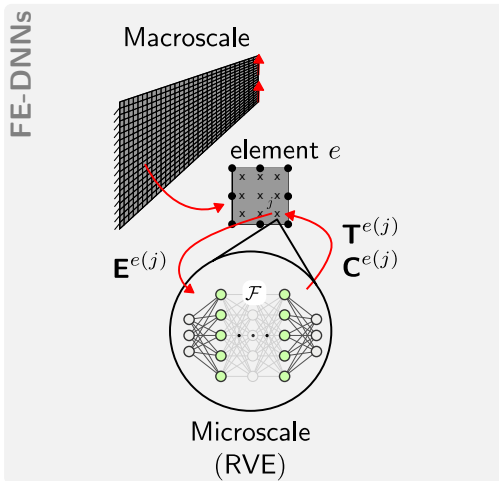
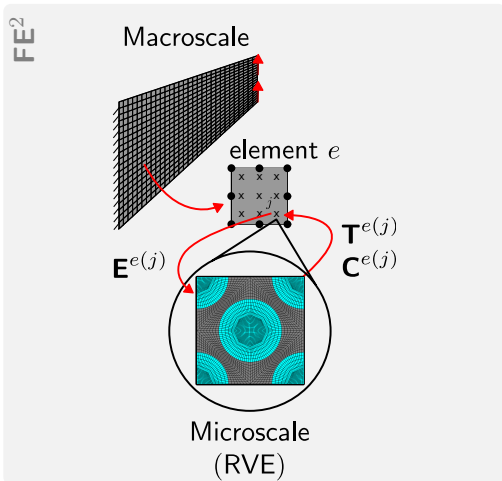
Results and discussion

Summary and conclusion

FE² computations with data-driven surrogates

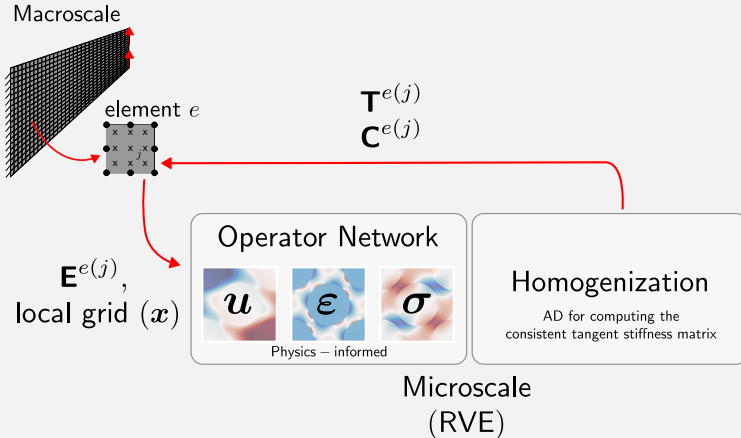


FE² computations with data-driven surrogates

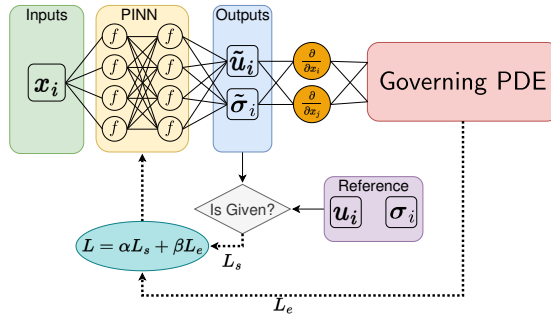


Our contribution: Operator learning for multiscale simulations

FE-DeepONet



Physics-informed neural networks (PINNs)



$$\frac{\partial u_i}{\partial x_j} = \mathcal{G}(u_i, x_j)$$

\mathcal{G} indicates chain-rule operation.

PINNs for continuum micromechanics and heterogeneous domains

Let us consider PDEs for elastostatics; we have Dirichlet and Neumann BCs

$$\mathbf{u}(\mathbf{x}) = \bar{\mathbf{u}}(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \quad (1)$$

$$\mathbf{t}(\mathbf{x}) = \boldsymbol{\sigma}(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) = \bar{\mathbf{t}}(\mathbf{x}) = \bar{\boldsymbol{\sigma}}(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \quad (2)$$

PINNs for continuum micromechanics and heterogeneous domains

Let us consider PDEs for elastostatics; we have Dirichlet and Neumann BCs

$$\mathbf{u}(\mathbf{x}) = \bar{\mathbf{u}}(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \quad (1)$$

$$\mathbf{t}(\mathbf{x}) = \boldsymbol{\sigma}(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) = \bar{\mathbf{t}}(\mathbf{x}) = \bar{\boldsymbol{\sigma}}(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \quad (2)$$

and balance law of linear momentum, kinematic relation and constitutive relation

$$\nabla \cdot \boldsymbol{\sigma}(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega, \quad (3)$$

$$\boldsymbol{\varepsilon}(\mathbf{x}) = \frac{1}{2}(\nabla \mathbf{u}(\mathbf{x}) + \nabla \mathbf{u}^T(\mathbf{x})), \quad \mathbf{x} \in \Omega, \quad (4)$$

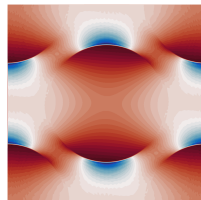
$$\boldsymbol{\sigma}(\mathbf{x}) = \mathcal{C}(\boldsymbol{\varepsilon})(\mathbf{x}), \quad \mathbf{x} \in \Omega. \quad (5)$$

PINNs for continuum micromechanics and heterogeneous domains

Problems for inhomogeneous domains:

- Numerical problems for sharp material phase transitions
- Second derivatives of $u(x)$ are needed.

$\sigma(x)$



¹A. Henkes et al. (2022)

²E. Samaniego et al. (2020)

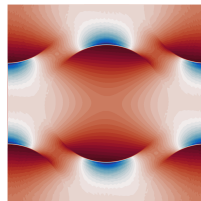
PINNs for continuum micromechanics and heterogeneous domains

Problems for inhomogeneous domains:

- Numerical problems for sharp material phase transitions
- Second derivatives of $\mathbf{u}(\mathbf{x})$ are needed.

We let the network output both $\mathbf{u}(\mathbf{x})$ and $\boldsymbol{\sigma}(\mathbf{x})$ ¹.
Alternately, deep energy method (DEM)² could be used.

$\boldsymbol{\sigma}(\mathbf{x})$

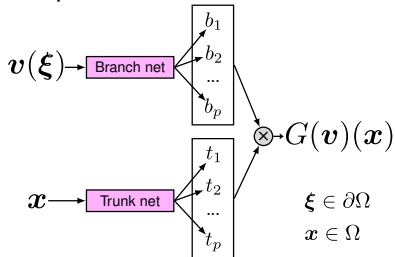


¹A. Henkes et al. (2022)

²E. Samaniego et al. (2020)

Deep operator networks (DeepONets)

DeepONets^a



automatically learn the basis.

^aLu Lu et al. (2020)

Maps between function spaces

$$G : v \mapsto G(v).$$

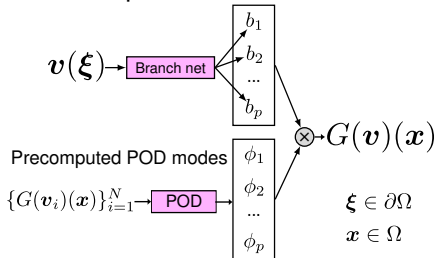
The output can be written as

$$G(v)(x) \approx \sum_{k=1}^p b_k(v) t_k(x) + b_0.$$

- t_k : a set of basis functions
- b_k : coefficients of the basis

Proper orthogonal decomposition (POD)-DeepONets

POD-DeepONets^a



employ precomputed POD modes.

^aLu Lu et al. (2022)

Maps between function spaces

$$G : v \mapsto G(v).$$

The output can be written as

$$G(v)(x) \approx \sum_{k=1}^p b_k(v) \phi_k(x) + \phi_0(x).$$

- t_k : ϕ_k : a set of POD basis
- b_k : coefficients of the basis
- ϕ_0 is the mean function.

Physics-informed DeepONets

Partial derivatives of the output

$$\frac{\partial G(\mathbf{v})(\mathbf{x})}{\partial x_i} \approx \sum_{k=1}^p b_k(\mathbf{v}) \frac{\partial \phi_k(\mathbf{x})}{\partial x_i} + \frac{\partial \phi_0(\mathbf{x})}{\partial x_i}, \quad i = 1, \dots, d.$$

Physics-informed DeepONets

Partial derivatives of the output

$$\frac{\partial G(\mathbf{v})(\mathbf{x})}{\partial x_i} \approx \sum_{k=1}^p b_k(\mathbf{v}) \frac{\partial \phi_k(\mathbf{x})}{\partial x_i} + \frac{\partial \phi_0(\mathbf{x})}{\partial x_i}, \quad i = 1, \dots, d.$$

DeepONet

Automatic differentiation

$$\frac{\partial \phi_i}{\partial x_j} = \mathcal{G}(\phi_i, x_j)$$

Physics-informed DeepONets

Partial derivatives of the output

$$\frac{\partial G(\mathbf{v})(\mathbf{x})}{\partial x_i} \approx \sum_{k=1}^p b_k(\mathbf{v}) \frac{\partial \phi_k(\mathbf{x})}{\partial x_i} + \frac{\partial \phi_0(\mathbf{x})}{\partial x_i}, \quad i = 1, \dots, d.$$

DeepONet

Automatic differentiation

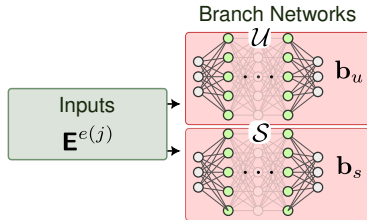
$$\frac{\partial \phi_i}{\partial x_j} = \mathcal{G}(\phi_i, x_j)$$

POD-DeepONet

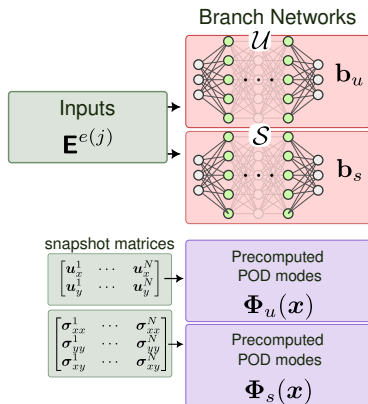
Numerical differentiation

$$\frac{\partial \phi_i}{\partial x_j} = \sum_{k=1}^n \phi_{ik} \frac{\partial N_k}{\partial x_j}$$

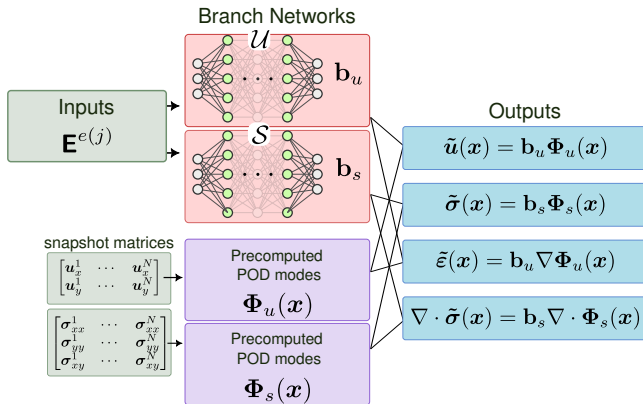
FE-DeepONet architecture



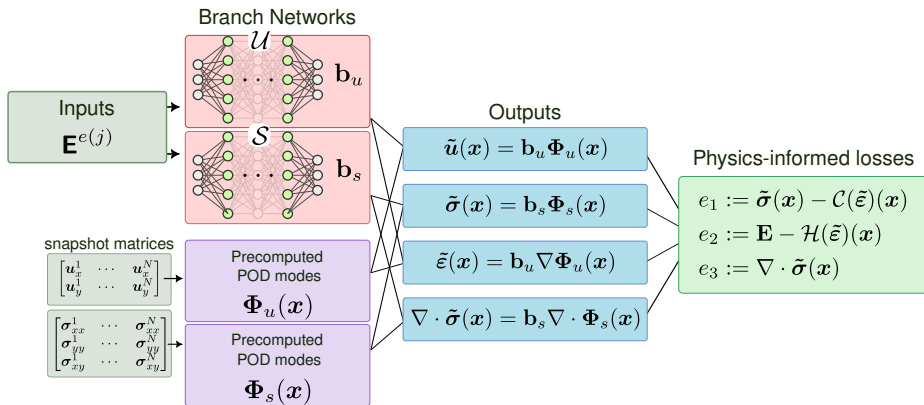
FE-DeepONet architecture



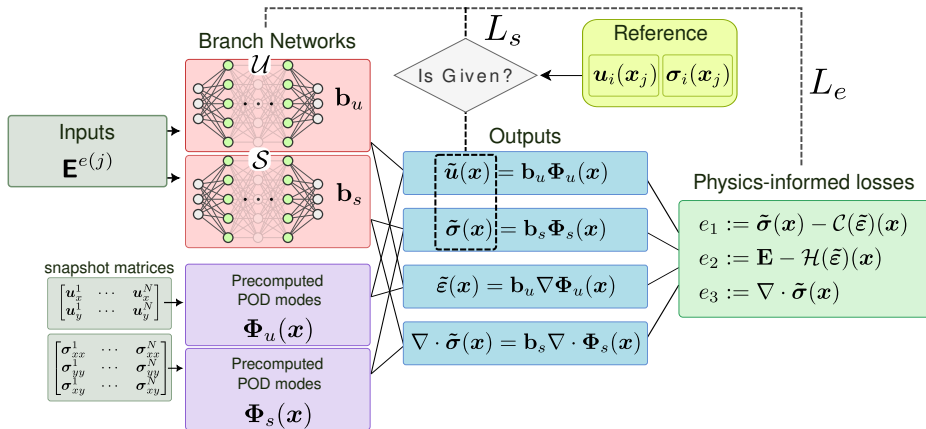
FE-DeepONet architecture



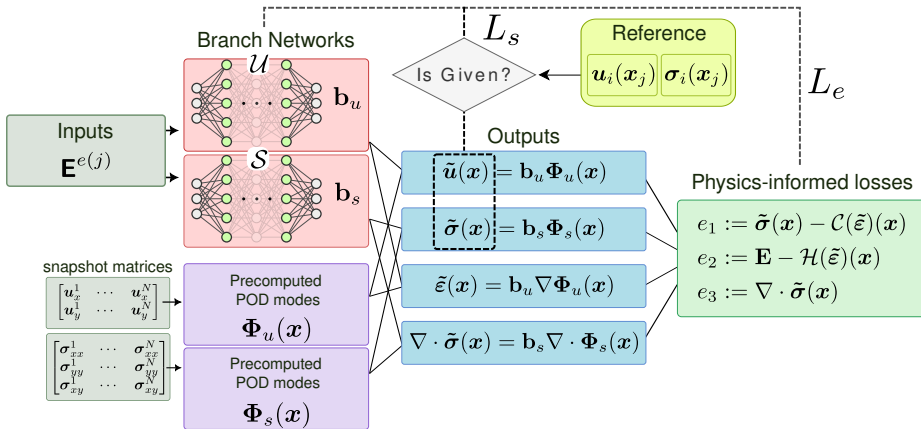
FE-DeepONet architecture



FE-DeepONet architecture



FE-DeepONet architecture



Similar to POD-Galerkin Reduced Order Methods.

FE-DeepONet loss function

Supervised losses:

$$L_u = \frac{1}{N^s} \sum_{i=1}^{N^s} \|\mathbf{u}_i(\mathbf{x}) - \tilde{\mathbf{u}}_i(\mathbf{x})\|_2^2$$

$$L_\sigma = \frac{1}{N^s} \sum_{i=1}^{N^s} \|\boldsymbol{\sigma}_i(\mathbf{x}) - \tilde{\boldsymbol{\sigma}}_i(\mathbf{x})\|_2^2$$

$$\mathbf{x} \in \partial\Omega \text{ or } \Omega$$

FE-DeepONet loss function

Supervised losses:

$$L_u = \frac{1}{N^s} \sum_{i=1}^{N^s} \|\mathbf{u}_i(\mathbf{x}) - \tilde{\mathbf{u}}_i(\mathbf{x})\|_2^2$$

$$L_\sigma = \frac{1}{N^s} \sum_{i=1}^{N^s} \|\boldsymbol{\sigma}_i(\mathbf{x}) - \tilde{\boldsymbol{\sigma}}_i(\mathbf{x})\|_2^2$$

$\mathbf{x} \in \partial\Omega$ or Ω

Unsupervised losses:

$$L_{\text{const.}} = \frac{1}{N^e} \sum_{i=1}^{N^e} \|\tilde{\boldsymbol{\sigma}}_i(\mathbf{x}) - \mathcal{C}(\tilde{\boldsymbol{\varepsilon}})_i(\mathbf{x})\|_2^2$$

$$L_{\text{homog.}} = \frac{1}{N^e} \sum_{i=1}^{N^e} \|\mathbf{E}_i - \mathcal{H}(\tilde{\boldsymbol{\varepsilon}})_i(\mathbf{x})\|_2^2$$

$$L_{\text{momen.}} = \frac{1}{N^e} \sum_{i=1}^{N^e} \|\nabla \cdot \tilde{\boldsymbol{\sigma}}_i(\mathbf{x})\|_2^2$$

FE-DeepONet loss function

Supervised losses:

$$L_u = \frac{1}{N^s} \sum_{i=1}^{N^s} \|\mathbf{u}_i(\mathbf{x}) - \tilde{\mathbf{u}}_i(\mathbf{x})\|_2^2$$

$$L_\sigma = \frac{1}{N^s} \sum_{i=1}^{N^s} \|\boldsymbol{\sigma}_i(\mathbf{x}) - \tilde{\boldsymbol{\sigma}}_i(\mathbf{x})\|_2^2$$

$\mathbf{x} \in \partial\Omega$ or Ω

Unsupervised losses:

$$L_{\text{const.}} = \frac{1}{N^e} \sum_{i=1}^{N^e} \|\tilde{\boldsymbol{\sigma}}_i(\mathbf{x}) - \mathcal{C}(\tilde{\boldsymbol{\varepsilon}})_i(\mathbf{x})\|_2^2$$

$$L_{\text{homog.}} = \frac{1}{N^e} \sum_{i=1}^{N^e} \|\mathbf{E}_i - \mathcal{H}(\tilde{\boldsymbol{\varepsilon}})_i(\mathbf{x})\|_2^2$$

$$L_{\text{momen.}} = \frac{1}{N^e} \sum_{i=1}^{N^e} \|\nabla \cdot \tilde{\boldsymbol{\sigma}}_i(\mathbf{x})\|_2^2$$

Total loss: $L = \alpha_1 L_u + \alpha_2 L_\sigma + \alpha_3 L_{\text{const.}} + \alpha_4 L_{\text{homog.}} + \alpha_5 L_{\text{momen.}}$

FE-DeepONet training

Limited dataset: only 100 samples

Data Generation

- 100 samples
- Latin hypercube sampling (LHS)

FE-DeepONet training

Limited dataset: only 100 samples

Data Generation

- 100 samples
- Latin hypercube sampling (LHS)

POD

- Snapshot matrices
- SVD
- Select $r = 16$ modes
- Compute mode derivatives

FE-DeepONet training

Limited dataset: only 100 samples

Data Generation

- 100 samples
- Latin hypercube sampling (LHS)

POD

- Snapshot matrices
- SVD
- Select $r = 16$ modes
- Compute mode derivatives

Training

- N^s supervised
- N^e unsupervised
- Adam 1e4 epochs
- L-BFGS-B
fine-tuning until
convergence

Multiscale simulation: Setup Representative volume element (RVE)

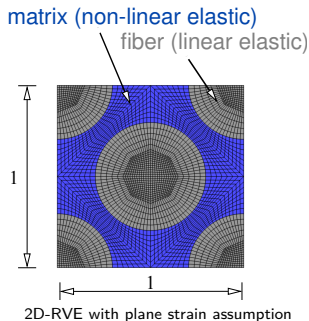
Non-linear elasticity:

$$\boldsymbol{\sigma} = K_m (\text{tr } \boldsymbol{\varepsilon}) \mathbf{I} + G_m (\boldsymbol{\varepsilon}^D) \boldsymbol{\varepsilon}^D,$$

$$G_m (\boldsymbol{\varepsilon}^D) = \frac{\alpha_1}{\alpha_2 + \|\boldsymbol{\varepsilon}^D\|_2}.$$

Linear elasticity:

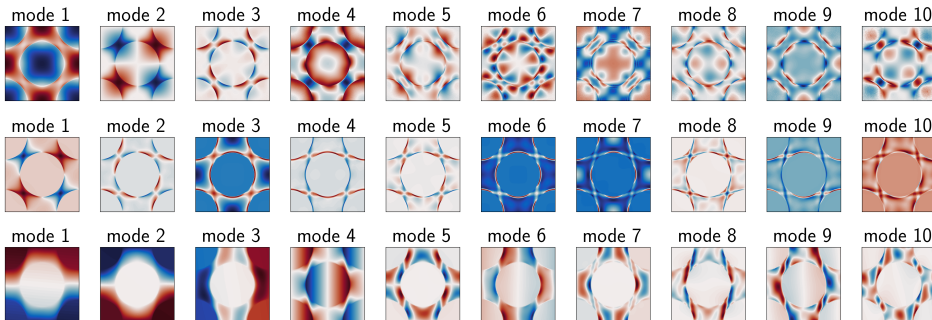
$$\boldsymbol{\sigma} = K_f (\text{tr } \boldsymbol{\varepsilon}) \mathbf{I} + 2G_f (\boldsymbol{\varepsilon}^D).$$



K_f N mm^{-2}	G_f N mm^{-2}	K_m N mm^{-2}	α_1 N mm^{-2}	α_2 -
4.35×10^4	2.99×10^4	4.78×10^3	5.0×10^1	6.0×10^{-2}

FE-DeepONet training POD modes

The first 10 POD modes of σ_{xy} , ε_{xy} , and u_y



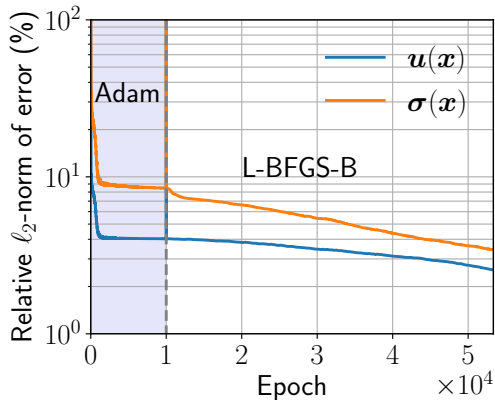


FE-DeepONet training Physics-informed training

Training history for $N^s = 0$ and $N^e = 1,000$:

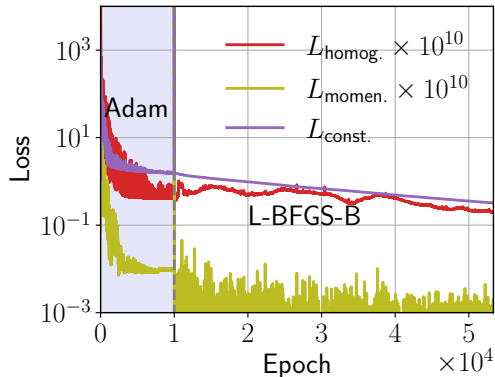
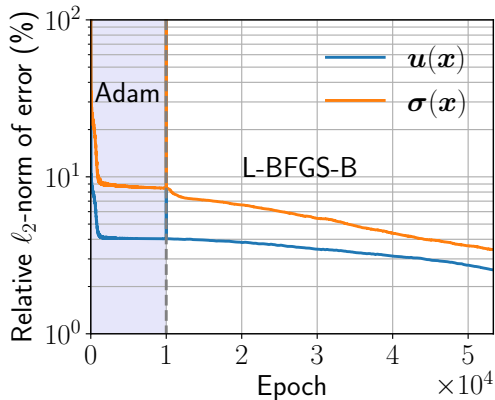
FE-DeepONet training Physics-informed training

Training history for $N^s = 0$ and $N^e = 1,000$:



FE-DeepONet training Physics-informed training

Training history for $N^s = 0$ and $N^e = 1,000$:

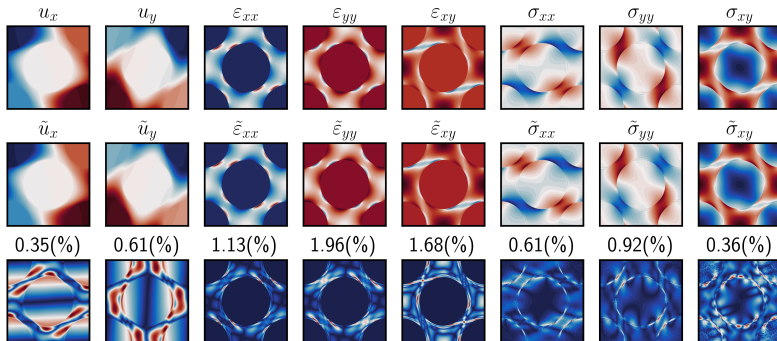


FE-DeepONet training Physics-informed training

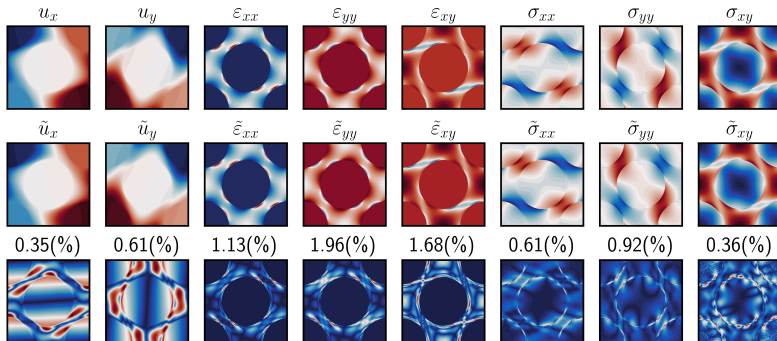
Balance law of linear momentum is satisfied by construct.

$$\begin{aligned}\text{From training data } \nabla \cdot \boldsymbol{\sigma}(\boldsymbol{x}) = \mathbf{b}_s \nabla \cdot \boldsymbol{\Phi}_s(\boldsymbol{x}) \approx 0 &\Rightarrow \nabla \cdot \boldsymbol{\Phi}_s(\boldsymbol{x}) \approx 0, \\ \nabla \cdot \tilde{\boldsymbol{\sigma}}(\boldsymbol{x}) = \tilde{\mathbf{b}}_s \nabla \cdot \boldsymbol{\Phi}_s(\boldsymbol{x}) &\approx 0.\end{aligned}$$

RVE simulation



RVE simulation



Relative ℓ_2 -norm of error

N^s	\mathbf{u}	$\boldsymbol{\varepsilon}$	$\boldsymbol{\sigma}$
0	2.82%	7.06%	3.75%
100	0.78%	3.06%	1.11%

Multiscale simulation

Forward map of the network for multiscale simulation

Macroscale stress

$$\mathbf{T} = f(\mathbf{E}; \theta_s, \Phi_s, \mathbf{x}) = \mathcal{H}(\mathcal{S}(\mathbf{E}) \cdot \Phi_s)(\mathbf{x}),$$
$$\mathbf{T} = \mathbf{b}_s \mathcal{H}(\Phi_s)(\mathbf{x}) = \mathbf{b}_s \frac{\sum_{j=1}^{N_i} w_j \Phi_s(\mathbf{x}_j)}{\sum_{j=1}^{N_i} w_j} \Rightarrow (b, r) \times (r, d),$$

Multiscale simulation

Forward map of the network for multiscale simulation

Macroscale stress

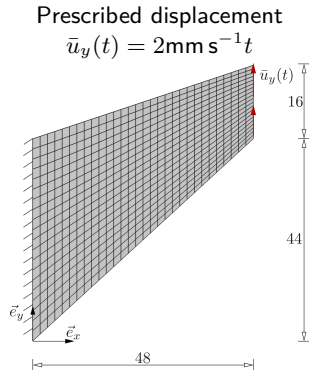
$$\mathbf{T} = f(\mathbf{E}; \theta_s, \Phi_s, \mathbf{x}) = \mathcal{H}(\mathcal{S}(\mathbf{E}) \cdot \Phi_s)(\mathbf{x}),$$
$$\mathbf{T} = \mathbf{b}_s \mathcal{H}(\Phi_s)(\mathbf{x}) = \mathbf{b}_s \frac{\sum_{j=1}^{N_i} w_j \Phi_s(\mathbf{x}_j)}{\sum_{j=1}^{N_i} w_j} \Rightarrow (b, r) \times (r, d),$$

Consistent tangent stiffness matrix

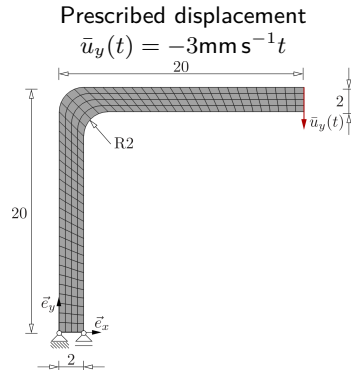
$$\mathbf{C} = \frac{df(\mathbf{E}; \theta_s, \Phi_s, \mathbf{x})}{d\mathbf{E}}.$$

Multiscale simulation: Test cases

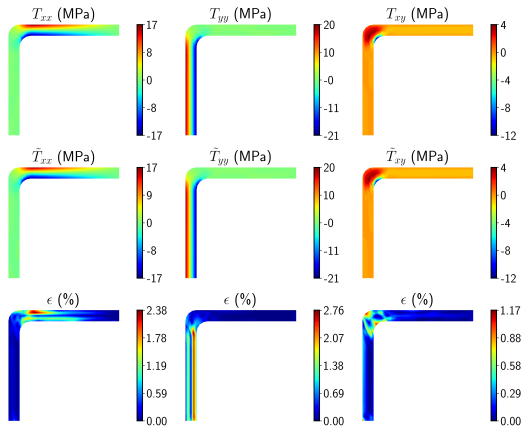
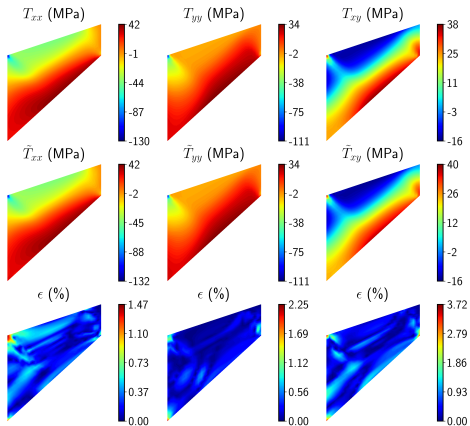
Cook's membrane:



L-profile:



Multiscale simulation: Results



Multiscale simulation: Results

Relative ℓ_2 -norm of errors					
Test	u_x	u_y	σ_{xx}	σ_{yy}	σ_{xy}
Cook's	0.3%	0.22%	1.44%	1.62%	1.28%
L-profile	0.88%	0.25%	2.53%	2.8%	2.57%

Multiscale simulation: Results

Relative ℓ_2 -norm of errors					
Test	u_x	u_y	σ_{xx}	σ_{yy}	σ_{xy}
Cook's	0.3%	0.22%	1.44%	1.62%	1.28%
L-profile	0.88%	0.25%	2.53%	2.8%	2.57%

- 3000x speed-up

Multiscale simulation: Results

Relative ℓ_2 -norm of errors					
Test	u_x	u_y	σ_{xx}	σ_{yy}	σ_{xy}
Cook's	0.3%	0.22%	1.44%	1.62%	1.28%
L-profile	0.88%	0.25%	2.53%	2.8%	2.57%

- 3000x speed-up
- 2 hours training time on a GPU

Summary and conclusion

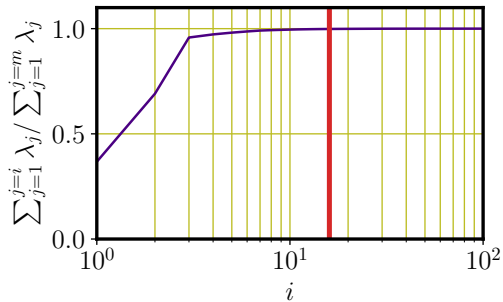
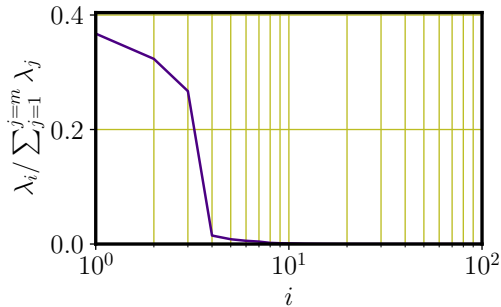
- A complementary model rather than a substitute
- First hybrid FE operator-learning solver for multiscale simulations
- Physics-informed deep operator networks for microscale simulation
- Only needs a limited number of training samples
- Balance law of linear momentum is satisfied by construct
- More than three orders of magnitude speed-up



Thank you!
Any questions?

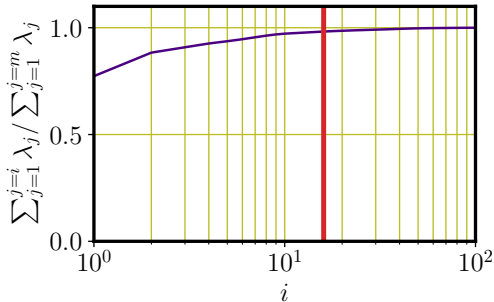
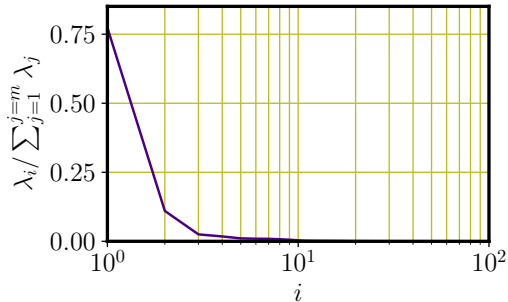
FE-DeepONet training POD truncation

Truncation of the modes for u



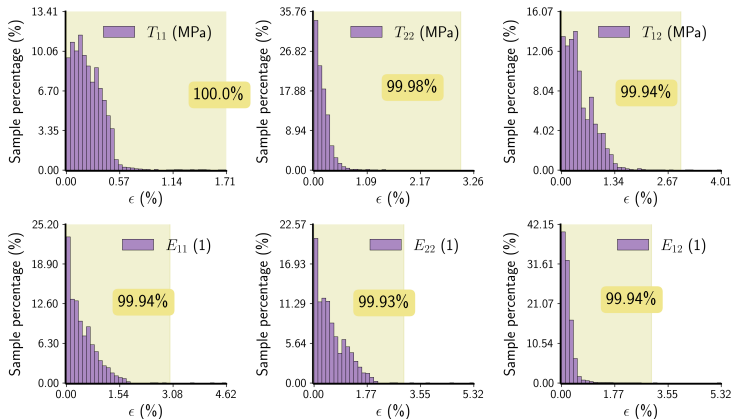
FE-DeepONet training POD truncation

Truncation of the modes for σ



Multiscale simulation: Results

Cook's membrane



Multiscale simulation: Results

L-profile

